Algorithms and Hardness for Non-Pool-Based Line Planning

Irene Heinrich
TU Darmstadt, Germany

Philine Schiewe
TU Kaiserslautern, Germany

Constantin Seebach
TU Kaiserslautern, Germany

Abstract

Line planning, i.e. choosing paths which are operated by one vehicle end-to-end, is an important aspect of public transport planning. While there exist heuristic procedures for generating lines from scratch, most theoretical observations consider the problem of choosing lines from a predefined line pool. In this paper, we consider the complexity of the line planning problem when all simple paths can be used as lines. Depending on the cost structure, we show that the problem can be NP-hard even for paths and stars, and that no polynomial time approximation of sub-linear performance is possible. Additionally, we identify polynomially solvable cases and present a pseudo-polynomial solution approach for trees.

2012 ACM Subject Classification Applied computing → Transportation; Mathematics of computing → Discrete optimization; Theory of computation → Problems, reductions and completeness; Theory of computation → Discrete optimization; Theory of computation → Design and analysis of algorithms

Keywords and phrases line planning, public transport, discrete optimization, complexity, algorithm design

Digital Object Identifier 10.4230/OASIcs.ATMOS.2022.8

Funding The research leading to these results has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (EngageS: grant agreement No. 820148) and by DFG under SCHO 1140/8-2.

1 Introduction

In public transport planning, *lines* are crucial building blocks. As lines are (simple) paths in the public transport network that have to be covered by one vehicle end-to-end, they highly influence the subsequent steps like timetabling and vehicle scheduling, see [14] and Figure 1. On the one hand, lines influence the passengers by providing routes and transfers and on the other hand, they determine the majority of the operating costs. Thus, line planning is an important foundation for building a public transport supply. From a set of lines, the *line pool*, a subset of lines and their frequencies, called *line concept*, is chosen for operation. The frequency of a line determines how often it is operated per planning period. While there is ample literature on line planning for a given fixed line pool, see [22], the construction of line pools is often neglected. In this paper, we focus on designing line concepts without a given line pool. Instead, we consider the set of all simple paths as candidates thus extending the solution space in the *line planning on all lines problem* LPAL. We show that, assuming $P \neq NP$, polynomial time approximations cannot give a performance guarantee that is better than linear and that depending on the cost-structure, the problem is NP-hard even for simple graph classes. Additionally, we identify polynomially solvable cases and develop a pseudo-polynomial algorithm for trees.

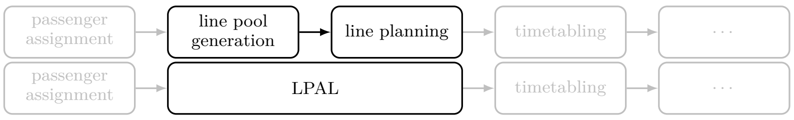

Figure 1 Sequential approach for public transport planning, adapted from [14] and integrated version considered here.

Literature review. Traditionally, the process of line planning is split into two stages, see Figure 1. For a given passenger assignment, a line pool is constructed and for this fixed pool, a line concept is determined. Many (meta-)heuristic approaches exist for the transit network design problem, where lines and often also passenger routes are generated [17, 9]. Usually, lines are supposed to not deviate too much from shortest paths [1, 6], or a set of lines to choose from is precomputed [26].

There is ample literature on line planning for a given line pool, i.e. a set from which lines are chosen for operation, see [22]. The most important objectives for passengers are to maximize the number of direct travelers [5] or to minimize the travel time [23, 4]. Here, it is especially difficult to model passenger behavior realistically, see [13, 21].

In this paper we focus on minimizing the costs of a line concept as originally introduced in [7]. By assigning passenger routes in a previous step, see Figure 1, it is guaranteed that passengers can travel on favorable routes, see e.g. [5]. As in [27, 24, 25], we distinguish between frequency-dependent and frequency-independent costs. Frequency-dependent costs can include costs for the distance covered by the lines and for the number of vehicles needed to operate the given line plan while frequency-independent costs can e.g. be used to reduce the number of different lines operated.

When solving the line planning problem for a fixed line pool, the line pool has a large influence on the complexity of the problem and the quality of solutions. One approach is to handle the generation of a suitable pool as an optimization problem itself, see [12]. Another possibility is to solve the line planning problem on the set of all possible lines. This idea has been studied using a column-generation approach in [3] where lines are only allowed to start and end at terminal stations. For this case, the line planning problem was shown to be NP-hard on planar graphs. See [2, 25] for further results on the complexity of the problem using terminal stations in path networks representing Istanbul Metrobüs and trees representing the Quito Trolebús. In [19], an integer programming formulation which includes line planning on all lines is presented and applied to small instances while in [18], line planning on all *circular* lines in a specific class of graphs is considered.

Our contribution. We focus on the line planning problem on all simple paths depending on the cost structure and show that the problem is hard to solve approximately in polynomial time: a sub-linear approximation ratio would imply $P = NP$, and even with another simplification of the problem, no constant approximation ratio is possible unless $P = NP$. We show that this problem is NP-hard even on planar graphs both when considering only frequency-dependent costs and when considering frequency-independent costs. The inclusion of frequency-independent costs makes the problem NP-hard even on paths and stars. Considering only frequency-dependent costs, we identify both polynomially solvable and NP-hard cases. Additionally, we present a pseudo-polynomial algorithm for trees and a polynomial one for special cases. An overview of these results is presented in Table 1.

Table 1 Approximation hardness and complexity of LPAL assuming $P \neq NP$.

graph class	no frequency-independent costs ($d_{\text{fix}} = 0$)	with frequency-independent costs ($d_{\text{fix}} > 0$)
general	no polynomial-time $n^{1-\epsilon}$ approximation (Theorem 2) no polynomial-time constant factor approximation, even for $f^{\max} \equiv \infty$ (Theorem 3)	
planar	NP-hard, even for $\{0,1\}$ input (Corollary 5)	
paths	polynomial for $f^{\max} \equiv \infty$ (see [11])	NP-hard (Theorem 7)
stars	polynomial (Theorem 11)	NP-hard (Theorem 8)
trees	pseudo-polynomial (Theorem 12) polynomial for $f^{\min} = f^{\max}$ (Theorem 14)	NP-hard (Theorems 7 and 8)

Outline. In Section 2 we formally introduce the line planning on all lines problem. We discuss the hardness of approximation in Section 3 and prove new NP-hardness results in Section 4. Section 5 contains a polynomial algorithm for stars and in Section 6 we develop a pseudo-polynomial solution approach for trees as well as a polynomial version for a special case.

2 Preliminaries

Graph theory. All graphs in this paper are undirected, finite, simple and non-empty. Whenever we consider a graph $G = (V, E)$, we use $n := |V|$ to denote its number of vertices. We measure the complexity of graph problems dependent on n. The *degree* $\deg(v)$ of a vertex v is the number of its neighbors. A graph (V, E) with $V = \{v_1, \dots, v_m\}$ and $E = \{\{v_1, v_2\}, \dots, \{v_{m-1}, v_m\}\}$ where all the v_i are distinct is a simple v_1-v_m-*path* (or just *path*). Paths can be specified as a sequence of vertices or as a sequence of edges. A complete bipartite graph of the form $K_{1,k}$ is a *star*. A graph is *traceable* if it has a Hamiltonian path.

Line planning. A *public transport network (PTN)* is a graph $G = (V, E)$ whose vertices represent stations while its edges represent direct connections between the stations, e.g., streets or tracks. A *line planning instance* is a tuple $(G, d_{\text{fix}}, c_{\text{fix}}, c, f^{\min}, f^{\max})$, where

- $G = (V, E)$ is a PTN,
- $d_{\text{fix}} \in \mathbb{R}_{\geq 0}$ represents *frequency-independent fixed costs*,
- $c_{\text{fix}} \in \mathbb{R}_{\geq 0}$ represents *frequency-dependent fixed costs*,
- $c \colon E \to \mathbb{R}_{\geq 0}$, $e \mapsto c_e$ is a map representing the *edge-dependent costs*, and
- $f^{\min}$ and $f^{\max} \colon E \to \mathbb{N}$ are *integer frequency restrictions* on E, $e \mapsto f_e^{\min}$ (respectively $e \mapsto f_e^{\max}$) such that $f_e^{\min} \leq f_e^{\max}$ for all edges $e \in E$. Note that the lower frequency restrictions $f_e^{\min}$ allow for passengers traveling on favorable routes while the upper frequency restrictions $f_e^{\max}$ represent safety constraints.

A *line* ℓ is a simple path in G and a *line concept* $(\mathcal{L}, f)$ is a set of lines $\mathcal{L}$ with a *frequency vector* $f = (f_\ell)_{\ell \in \mathcal{L}} \in \mathbb{N}^{|\mathcal{L}|}$, i.e. f_ℓ is the *frequency* of line ℓ. At each edge $e \in E$, the lines sum up to a total frequency

$$F_e^{(\mathcal{L},f)} = \sum_{\ell \in \mathcal{L} \colon e \in E(\ell)} f_\ell,$$

where $E(\ell)$ denotes the edge set of ℓ. We say that an edge e is *covered* by a line ℓ if $e \in E(\ell)$. A line concept is *feasible* if for each edge $e \in E$ the frequency restrictions are satisfied, i.e.

$f_e^{\min} \leq F_e^{(\mathcal{L},f)} \leq f_e^{\max}$. The set of feasible line concepts is $\mathcal{F}(G, f^{\min}, f^{\max})$ which we may abbreviate by writing $\mathcal{F}(G)$.

We use frequency-dependent *line costs* $\text{cost}_\ell = c_{\text{fix}} + \sum_{e \in E(\ell)} c_e$ which consist of fixed costs c_{fix} and edge-dependent costs c_e, $e \in E$. Additionally, we use frequency-independent costs d_{fix} per line. We define the *costs of a line concept* $(\mathcal{L}, f)$ as

$$\text{cost}((\mathcal{L}, f)) = d_{\text{fix}} \cdot |\mathcal{L}| + \sum_{\ell \in \mathcal{L}} \text{cost}_\ell \cdot f_\ell.$$

With this notation, we can formally define the line planning on all lines problem.

▶ **Definition 1** (LPAL). *Given a line planning instance, the* line planning on all lines problem LPAL *is to find a feasible line concept with minimal costs.*

3 Hardness of approximation

In this section we show that even in the case $d_{\text{fix}} = 0$, no polynomial time approximation algorithm for LPAL has a sub-linear performance ratio. Even when additionally $f^{\max} \equiv \infty$, no constant-factor polynomial time approximation is possible.

▶ **Theorem 2.** *Assuming $P \neq NP$, the problem* LPAL *cannot be approximated within a factor of $n^{1-\epsilon}$ by a polynomial-time algorithm, even in the case $d_{fix} = 0$.*

Proof. We prove this using a gap-producing reduction from the NP-complete problem Hamiltonian Path (see [10]).

Consider a directed graph G on n vertices. We claim that a line planning instance $I = (G', d_{\text{fix}}, c_{\text{fix}}, c, f^{\min}, f^{\max})$ with $d_{\text{fix}} = 0$, $c_{\text{fix}} = 1$ and $c \equiv 0$ can be constructed from G in polynomial time, where two special vertices v_1 and v_2 are marked, and it holds:

- If G is traceable: I can be solved using a single line with endings v_1 and v_2.
- If G is not traceable: I requires at least two lines.

To construct I, we first apply the reduction from [12] to G, creating an LPAL instance that can be solved using a single line if and only if G is traceable. Then we join two new vertices u_1 and u_2 to all other vertices with edges having $f_e^{\min} = 0$. To u_1 we join a new vertex v_1, likewise we join a new vertex v_2 to u_2 using edges having $f_e^{\min} = 1$. This forces a line from v_1 to v_2, but otherwise preserves the reduction equivalence.

Starting from G', for every $k \in \mathbb{N}_{\geq 1}$, we construct a graph G_k as follows: create k copies of G', and for all $i \in [1, k-1]$, add an edge between v_2 of copy i and v_1 of copy $i+1$. Call these $k-1$ new edges *connectors*. Then G_k consists of kn vertices. Consider a new line planning instance I_k on G_k, where we also copied the weights from I onto G_k, and have $f_e^{\max} = 1$ on the connectors. If G is traceable, then I_k can be solved using a single line, which we get by concatenating the lines for each copy of G', with help of the connectors.

We say a line ℓ *visits* copy i if the vertices of ℓ and the i-th copy of G' intersect. If G is not traceable, then each copy of G' is visited by at least two different lines, totaling at least $2k$ visits. A single line can visit multiple copies of G' and crosses a connector for each additional visit. Since $f_e^{\max} = 1$, every connector can only be crossed once. This affords us $k-1$ visits. The remaining $k+1$ visits are paid by different lines, i.e. every line concept $\mathcal{L}$ solving I_k needs $|\mathcal{L}| \geq k+1$.

Now assume that for some $\epsilon \in (0, 1]$ we can approximate LPAL within $n^{1-\epsilon}$ using an algorithm A in polynomial time. We set $k := \lfloor 1 + n^{(1-\epsilon)/\epsilon} \rfloor$, i.e. k is the smallest integer larger than $n^{(1-\epsilon)/\epsilon}$. Given a graph G as input to Hamiltonian Path, we construct I_k, which has size kn, which is bounded by a polynomial in n. Then apply algorithm A, to get an approximate solution of cost a. If G is traceable, then the optimal solution to I_k has value 1. Hence $a \leq 1 \cdot (kn)^{1-\epsilon}$ and $a/k \leq n^{1-\epsilon}k^{-\epsilon} < n^{1-\epsilon} \cdot (n^{(1-\epsilon)/\epsilon})^{-\epsilon} = 1$. Thus $a < k$. If G is not traceable, then the optimal solution to I_k has value at least k, hence also $a \geq k$. By comparing a to k, we can determine whether G is traceable in polynomial time, implying $P = NP$. ◀

Since an n-approximation (or worse) for LPAL is useless in practice, we want to weaken the lower bound by putting more restrictions on the considered instances. In the preceding hardness proof, it was essential that we can use $f^{\max}$ to bound the frequencies.

In contrast we now consider instances, where $f^{\max} \equiv \infty$.

▶ **Theorem 3.** *Assuming $P \neq NP$, the problem* LPAL *cannot be approximated within a constant factor by a polynomial-time algorithm, even in the case $d_{fix} = 0$ and $f^{\max} \equiv \infty$.*

Proof. In this proof we assume all LPAL instances $(G = (V, E), d_{\text{fix}}, c_{\text{fix}}, c, f^{\min}, f^{\max})$ to have $f^{\max} \equiv \infty$, $d_{\text{fix}} = 0$, $c_{\text{fix}} = 1$, $c \equiv 0$, and $f_e^{\min} \in \{0, 1\}$ for all $e \in E$.

Let $I = (G = (V, E), d_{\text{fix}}, c_{\text{fix}}, c, f^{\min}, f^{\max})$ be an LPAL instance. An edge $e = \{v_1, v_2\} \in E$ with $f_e^{\min} = 1$ and $\deg(v_1) = 1$ is an *antenna* of I and v_1 is the *tip* of the antenna. We call I *nice* if it has exactly two antennae. Let p be a path on G and $V' \subseteq V$. The *restriction* of p to V' is the subgraph of p induced by $V' \cap V(p)$. The restriction is *proper* if it is a path.

If we add a new antenna to an instance I, then an optimal solution for the resulting instance needs at least as many lines as for I. If I can be solved using a single line, then it has at most two antennae. If I has exactly two antennae, then the single line has its ends at the antenna tips. However, if I has fewer than two antennae, we may attach new antennae until we have two, such that the resulting instance can still be solved using a single line.

Let I and J be nice instances. By abuse of notation, we define $I \times J$ to be an instance constructed in the following manner: replace every edge $e = \{u, v\}$ of J with $f_e^{\min} = 1$ by a copy of I and identify u and v with the antenna tips of that copy. For an example of $I \times J$, see Figure 2. We claim that $I \times J$ is also nice. In particular, the antennae of $I \times J$ are part

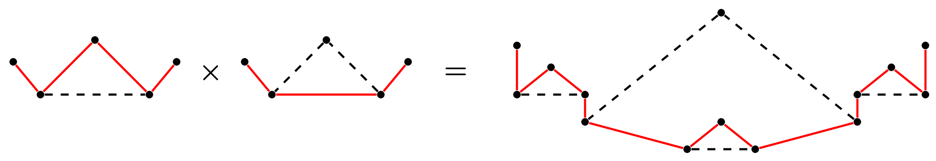

Figure 2 Example construction of $I \times J$. Edges with $f_e^{\min} = 1$ are red, other edges are dashed.

of two different copies of I. Denote these copies by $A^1_{I \times J}$ and $A^2_{I \times J}$, respectively. Let ℓ be a path on $I \times J$ and C be some copy of I which is part of $I \times J$. There are only two vertices where ℓ can enter or leave C. If ℓ starts outside C, it can enter C at most once. In that case, the restriction of ℓ to C is proper. If ℓ starts inside C, it may leave and enter again, which makes the restriction improper.

▷ Claim 4. $I \times I$ can be solved by a single line if and only if I can be solved by a single line. If an optimal solution for I requires $k \geq 2$ lines, then an optimal solution for $I \times I$ requires at least $k + 1$ lines.

Proof of Claim 4. First assume that I can be solved by the single line ℓ_I. Since I is nice the line ℓ_I ends in its antennae. By replacing every edge e of ℓ_I with $f_e^{\min} = 1$ by a copy of ℓ_I we obtain a line that solves $I \times I$.

Now assume that I requires $k \geq 2$ lines for an optimal solution. Suppose towards a contradiction that there are $k' \leq k$ lines $\ell_1, \ldots, \ell_{k'}$ that solve $I \times I$. Let $d \in \{1, 2\}$. Observe that $A^d_{I\times I}$ has the following property: one of its antennae is an antenna of $I \times I$, the other antenna is a bridge (or 1-edge-cut) of the underlying graph of $I \times I$. In particular, every line on $I \times I$ can be restricted properly to $A^d_{I\times I}$ and, hence, restricting all k' lines to $A^d_{I\times I}$ yields a feasible solution for $A^d_{I\times I}$. As $A^d_{I\times I}$ is a copy of I, $k = k'$ and every line ℓ_i has one end in $A^1_{I\times I}$ and the other in $A^2_{I\times I}$. Let I' be a copy of I which is inserted in the construction process of $I \times I$ and is neither $A^1_{I\times I}$ nor $A^2_{I\times I}$. Every line on $I \times I$ can be restricted properly to I' since the two antennae of I' form a 2-edge-cut of $I \times I$ and the line neither starts nor ends in I' by the above considerations. Since an optimal solution for I' requires k lines by assumption we obtain that every line ℓ_i for $i \in [1, k]$ intersects I'. Let $i \in [1, k]$. Since ℓ_i visits every copy of I which corresponds to an edge e of I with $f^{\min}_e = 1$, we can restrict ℓ_i to the vertices of I and obtain a path r on I, which visits all edges e with $f^{\min}_e = 1$. This implies that r solves I, which contradicts our assumption. $\triangleleft$

For an LPAL instance I and a number $k \in \mathbb{N}_{\geq 1}$ we define I^k as the repeated product $((I \times I) \times \ldots) \times I$ of k factors. If I can be solved using a single line, I^k can as well by Claim 1. Otherwise I^k requires at least $k + 1$ lines. The product I^k contains at most n^{2k} vertices.

Now assume that for some $\alpha \in [1, \infty)$, LPAL can be α-approximated using a polynomial time algorithm A. Set $k := \lfloor \alpha \rfloor$. We show how to decide Hamiltonian Path in polynomial time, implying $P = NP$.

Let G be a directed graph. Apply the reduction from [12] to G to obtain an LPAL instance I_0 that is solvable using a single line if and only if G is traceable. If I_0 has more than two antennae, then G is not traceable (each antenna corresponds to an end of a Hamiltonian path if one exists). If I_0 has two or fewer antennae, we consider all possible ways to attach antennae such that the constructed instance has exactly two antennae. This results in a list L of at most n^2 nice instances. If I_0 is solvable using a single line, then some instance $I \in L$ is, too. We repeat the following for every $I \in L$:

First construct I^k. This is possible in polynomial time since k does not depend on n. Apply A to I^k to obtain an approximately optimal line concept that has cost x. If I can be solved using one path, the minimal cost of solving I^k is 1. Hence $x \leq \alpha$. Otherwise the minimal cost of solving I^k is $k + 1$, hence $x \geq k + 1 > \alpha$. It follows that by comparing x to α, we can determine whether I can be solved using one path.

There is an $I \in L$ which can be solved using just one path if and only if G is traceable. ◀

As this hardness result is weaker, we could hope to find an approximation algorithm where the error grows only very slightly in n. This is an interesting open problem.

4 NP-hard cases

For general graphs and general cost structures, the problem of finding a cost-optimal line concept is known to be NP-hard, even if

- $d_{\text{fix}} = 1$, $c_{\text{fix}} = 0$, $c \equiv 0$, $f^{\min}_e \in \{0, 1\}$ for all $e \in E$, $f^{\max} \equiv \infty$ or $f^{\max} \equiv 1$ [12] or
- $d_{\text{fix}} = 0$, $f^{\min}_e \in \{0, 1\}$ for all $e \in E$, $f^{\max} \equiv \infty$ or $f^{\max} \equiv 1$ [11].

We can strengthen theses results and show that LPAL is NP-hard even for subcubic planar graphs.

▶ **Corollary 5.** *The problem* LPAL *is NP-hard, even if G is a planar graph with maximum vertex degree at most three and*
(a) $d_{\mathit{fix}} = 1$, $c_{\mathit{fix}} = 0$, $c \equiv 0$, $f^{\min}_e \in \{0, 1\}$ *for all* $e \in E$, $f^{\max} \equiv \infty$ *or* $f^{\max} \equiv 1$ *or*
(b) $d_{\mathit{fix}} = 0$, $f^{\min}_e \in \{0, 1\}$ *for all* $e \in E$, $f^{\max} \equiv \infty$ *or* $f^{\max} \equiv 1$.

Proof sketch. We combine the following two results:

1. In [11, 12] a reduction technique of HAMILTONIAN PATH to a problem equivalent to LPAL with restrictions (a) and (b), respectively, is presented. In general, this reduction does not preserve planarity.
2. Plesník [20] shows that HAMILTONIAN PATH is NP-hard even for planar digraphs where each vertex has in- and out-degree at most two and either in- or out-degree one.

By modifying the reductions of [11, 12] for these graphs, the constructed line planning instance consists of a planar graph with vertex degree at most three. ◀

In the remainder of this section we show that if frequency-independent costs d_{fix} are considered, then LPAL remains NP-hard even for paths and stars. To this end we formulate reductions that utilize $f^{\max}$. Lemma 6 can be applied to transfer the hardness results even if $f^{\max} \equiv \infty$.

▶ **Lemma 6** (Lifting $f^{\max}$). *Let $I = ((V, E), d_{fix}, c_{fix}, c, f^{\min}, f^{\max})$ be an instance to* LPAL *where $c_{fix} = 0$, $c \equiv 0$ and $f^{\min} = f^{\max}$. Let $K \in \mathbb{N}$. Define $I' := ((V, E), d_{fix}, c_{fix}, c', f^{\min}, \infty)$ with $c' :\equiv K + 1$ and $K' := K + (K + 1) \sum_{e\in E} f_e^{\min}$.*

Then I has a feasible line concept with cost at most K if and only if I' has a feasible line concept with cost at most K'. Both I' and K' can be computed in polynomial time.

Proof. We show that we can transfer a solution $(\mathcal{L}, f)$ from one instance to the other, such that it is still feasible and within the cost bound. We add a superscript to cost, to distinguish for which instance we view the costs.

$I \to I'$: Clearly, $(\mathcal{L}, f)$ remains feasible for I'. Since $c \equiv 0$ and $c_{\text{fix}} = 0$, we have $\text{cost}^I((\mathcal{L}, f)) = d_{\text{fix}} \cdot |\mathcal{L}|$, which is less or equal to K. Since $f^{\min} = f^{\max}$, the frequency-dependent line costs of I' are predetermined:

$$\begin{aligned}
\sum_{\ell\in\mathcal{L}} f_\ell \cdot \text{cost}_\ell^{I'} &= \sum_{\ell\in\mathcal{L}} f_\ell \cdot \left(c_{\text{fix}} + \sum_{e\in E(\ell)} c'_e \right) \\
&= \sum_{\ell\in\mathcal{L}} f_\ell \sum_{e\in E(\ell)} c'_e = \sum_{e\in E} c'_e \sum_{\substack{\ell\in\mathcal{L}:\\ e\in E(\ell)}} f_\ell = \sum_{e\in E} (K+1) f_e^{\min}
\end{aligned}$$

Then $\text{cost}^{I'}((\mathcal{L}, f)) = d_{\text{fix}} \cdot |\mathcal{L}| + \sum_{e\in E}(K + 1) f_e^{\min} \leq K + \sum_{e\in E}(K + 1) f_e^{\min} = K'$.

$I' \to I$: Towards a contradiction, assume $f_e^{\min} + 1 \leq \sum_{\ell\in\mathcal{L}:\, e\in E(\ell)} f_\ell$ for some $e \in E$. Then we derive in a similar fashion:

$$\begin{aligned}
\text{cost}^{I'}((\mathcal{L}, f)) = d_{\text{fix}} \cdot |\mathcal{L}| + \sum_{e\in E}(K+1) \sum_{\substack{\ell\in\mathcal{L}:\\ e\in E(\ell)}} f_\ell &\geq d_{\text{fix}} \cdot |\mathcal{L}| + (K+1) + \sum_{e\in E}(K+1) f_e^{\min} \\
&> K + (K+1) \sum_{e\in E} f_e^{\min} = K'
\end{aligned}$$

This contradicts $\text{cost}^{I'}((\mathcal{L}, f)) \leq K'$ and, hence, $f_e^{\min} = \sum_{\ell\in\mathcal{L}:\, e\in E(\ell)} f_\ell$ for all $e \in E$, implying that $(\mathcal{L}, f)$ is a feasible line concept for I. Subtracting the now fixed frequency-dependent line costs yields $\text{cost}^I((\mathcal{L}, f)) = d_{\text{fix}} \cdot |\mathcal{L}| \leq K$. ◀

First, we show that LPAL is NP-hard on paths.

▶ **Theorem 7.** *The problem* LPAL *is NP-hard, even if G is a path and $f^{\min} = f^{\max}$ or $f^{\max} \equiv \infty$.*

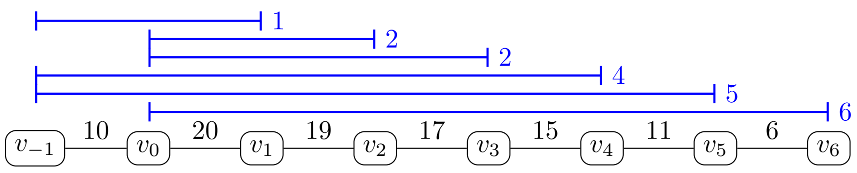

Figure 3 Example for the construction from Theorem 7, along with feasible line concept. Here $S = \{1, 2, 2, 4, 5, 6\}$.

Proof. We show a reduction of the NP-hard problem 3-Partition [10] to the decision version of LPAL, first for the case $f^{\min} = f^{\max}$. Let $S = \{x_1, \dots, x_{3p}\}$ be a multiset of positive integers. The idea of our construction is to have a path with one subpath of monotonically increasing frequency constraints and another subpath with monotonically decreasing frequency constraints. We call these subpaths *intervals* in reference to the interval of corresponding vertex indices. The first interval represents partitions $S_1, \dots, S_p$ while the second interval represents the elements of S. By choosing the frequency restrictions, we force the multiset of line frequencies to be exactly S. Then we can construct lines to have one end in the first interval and the other end in the second interval, representing to which set S_k an element $x_i \in S$ is assigned. In the first interval, lines can overlap in different ways, each representing a different way to partition S.

Define $h := \frac{1}{p} \sum S$. We may assume that h is integer and that every subset of S which sums to h contains exactly 3 elements as 3-Partition remains NP-hard in this case. Now define a sequence of integers, used for constructing the frequency restrictions:

$$a_i := \begin{cases} h & \text{if } i \le 0 \\ -x_i & \text{if } i > 0 \end{cases} \quad \text{for } i \in [1-p, 3p]. \text{ (Note that indices may be negative.)}$$

We construct our instance $I = (G, d_{\text{fix}}, c_{\text{fix}}, c, f^{\min}, f^{\max})$ with decision parameter K as follows:

- $d_{\text{fix}} := 1$
- $c_{\text{fix}} := 0$
- $c :\equiv 0$
- $K := 3p$.

The graph G is a path on $4p$ vertices, which we call $v_{1-p}, \dots, v_{3p}$. The edges are $e_i := \{v_i, v_{i+1}\}$ for $i \in [1-p, 3p-1]$. For all $i \in [1-p, 3p-1]$, we set $f^{\min}_{e_i} := f^{\max}_{e_i} := \sum_{j=1-p}^{i} a_j$. The construction is illustrated in Figure 3.

Consider a feasible solution $(\mathcal{L}, f)$ for I with $\operatorname{cost}((\mathcal{L}, f)) \le K$. From $d_{\text{fix}} = 1$ follows that $|\mathcal{L}| \le 3p$. Since G is a path, we can say that every line of $(\mathcal{L}, f)$ has a left and a right end. We first argue the case where the left end of each line $(v_i, \dots, v_j)$ is in the first interval, i.e. v_i satisfies $i \in [1-p, 0]$, and the right end is in the second interval, i.e. v_j satisfies $j \in [1, 3p]$.

For every $i \in [0, 3p-2]$ we have $F^{(\mathcal{L},f)}_{e_i} > F^{(\mathcal{L},f)}_{e_{i+1}}$, implying that at least one line has a right end at v_{i+1}. Also some line ends at v_{3p} since $F^{(\mathcal{L},f)}_{e_{3p-1}} = ph - \sum_{j=1}^{3p-1} x_i = x_{3p} > 0$. Hence $\mathcal{L}$ consists of exactly $3p$ lines, each having the right end at a different v_i for $i \in [1, 3p]$.

Let ℓ_i be the unique line ending at v_i. To make up the difference $F^{(\mathcal{L},f)}_{e_i} - F^{(\mathcal{L},f)}_{e_{i-1}} = x_i$ in G, we obtain $f_{\ell_i} = x_i$. For every $j \in [1-p, 0]$ consider the subset $\hat{L}_j$ of lines which have their left end at v_j. Their frequencies sum up to $F^{(\mathcal{L},f)}_{e_j} - F^{(\mathcal{L},f)}_{e_{j-1}} = a_j = h$. Since all lines have their left ends at some v_j with $j \in [1-p, 0]$, the sets $\hat{L}_{1-p}, \dots, \hat{L}_0$ partition $\mathcal{L}$ and correspond to a partition of S where each subset has sum h. This solves 3-Partition.

If there is a line whose left and right end are in the second interval, then it is no longer guaranteed that $f_{\ell_i} = x_i$ for the line ℓ_i with right end at v_i. Instead, f_{ℓ_i} exceeds x_i by the total frequency of lines whose left ends are at v_i. Now, we can elongate all lines with left end

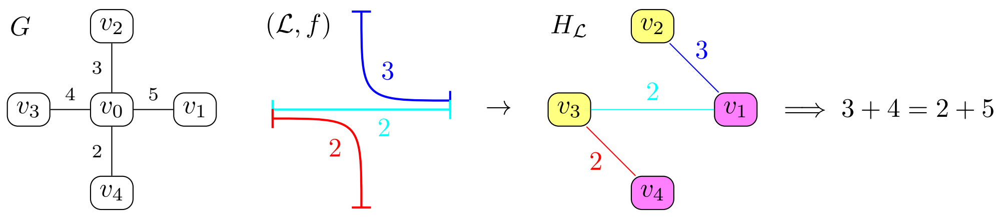

Figure 4 Example of the relationship between line concepts on stars and number partitions.

at v_i to the left end of ℓ_i and reduce the frequency of ℓ_i to x_i without introducing new lines or changing the total frequency of an edge. As there are no lines whose left end is v_{3p} and the right end of ℓ_1 has to be in the first interval, this allow us to construct a solution in the desired form in linear time.

Consider a solution $S_1 \cup \cdots \cup S_p = S$ to 3-PARTITION with $\sum S_k = h$ for all k. We construct a feasible line concept: for every $i \in [1, 3p]$, create a line ℓ_i with frequency x_i, having its right end at v_i. If $x_i \in S_k$, then ℓ_i has its left end at v_{1-k}. It is easy to check that these lines sum up exactly to the frequency profile of G, and the cost K is not exceeded.

To show hardness for the case $f^{\max} \equiv \infty$, we can apply Lemma 6. ◀

Additionally, LPAL is NP-hard on stars.

▶ **Theorem 8.** LPAL *is NP-hard, even if G is a star and $f^{\min} \equiv f^{\max}$ or $f^{\max} \equiv \infty$.*

In order to prove Theorem 8 we introduce the problem PARTITION INTO MANY PARTITIONS. First we prove that it is an NP-hard problem (Lemma 10) and then we reduce PARTITION INTO MANY PARTITIONS to the decision version of LPAL with the above assumptions.

▶ **Definition 9.** *PARTITION INTO MANY PARTITIONS is the following decision problem:*
Input: A set of positive integers S and a number K.
Question: Can a subset $S' \subseteq S$ be partitioned into at least K nonempty sets, such that each in turn is a yes-instance to PARTITION [16]?

▶ **Lemma 10.** *PARTITION INTO MANY PARTITIONS is strongly NP-hard.*

The proof of Lemma 10 is an reduction of PARTIAL LATIN SQUARE COMPLETION (which is shown to be NP-hard in [8]) to PARTITION INTO MANY PARTITIONS. It is an adaptation of the reduction described in [15] and can be found in Appendix A.

Before we present the full proof of Theorem 8 we provide a sketch of it in the following: consider a feasible line concept $(\mathcal{L}, f)$ on a star G where $f^{\min} = f^{\max}$ and the cost only includes the number of lines. We observe that whenever a one-edge line $\ell_1 \in \mathcal{L}$ shares an edge with some other line $\ell_2 \in \mathcal{L}$, we may obtain an equivalent line concept $\mathcal{L}'$ without edge-sharing by shortening ℓ_2 and increasing the frequency of ℓ_1 which may only reduce the cost. Hence, we may assume that only two-edge lines share an edge. We can visualize the edge intersection between lines as a graph $H_{\mathcal{L}}$ where each two-edge line (v_i, v_0, v_j) is represented by an edge $\{v_i, v_j\}$. If the resulting graph $H_{\mathcal{L}}$ has a cycle, then k edges are covered by k two-edge lines. We can therefore construct an equivalent line concept by covering each of the edges with a one-edge line and removing the corresponding edges from $H_{\mathcal{L}}$. Thus, we may assume that $H_{\mathcal{L}}$ is a forest, and we can use each tree component to obtain a number partition on some subset of the edge frequencies (see Figure 4). This makes the equivalence to PARTITION INTO MANY PARTITIONS apparent.

Proof of Theorem 8. We show a reduction of Partition into many Partitions to the decision version of LPAL. Let a set of positive integers $S = \{x_1, \dots, x_m\}$ and the lower bound K be given as an input to Partition into many Partitions.

We construct an instance $I = (G = (V, E), d_{\text{fix}}, c_{\text{fix}}, c, f^{\min}, f^{\max})$ with G and $f^{\min} = f^{\max}$ as in Figure 5 and decision parameter K' as follows: $d_{\text{fix}} := 1$, $c_{\text{fix}} := 0$, $c :\equiv 0$, and $K' := m - K$.

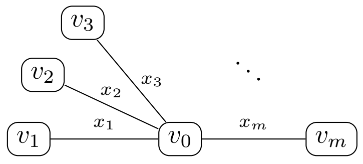

Figure 5 Line planning instance constructed in Theorem 8.

Let a solution $\text{Sol} = ((A_1, B_1), \dots, (A_K, B_K))$ to Partition into many Partitions be given, i.e. the sets $A_1, \dots, A_K, B_1, \dots, B_K$ are nonempty and form a partition of a subset of S with $\sum A_i = \sum B_i$ for all $i \in [1, K]$. We construct a solution to LPAL using Algorithm 1.

After each iteration of the for-loop starting in line 2, all edges $\{v_i, v_0\}$ corresponding to elements $x_i \in A \cup B$ are covered exactly $f^{\min}_{\{v_i,v_0\}}$ times. In each iteration of the while-loop starting in line 5, at least one element of A or B is removed and a new line is created. Note that $\sum A = \sum B$ is invariant as (A, B) is a partition such that the case in line 16 is reached. Thus, at most $|A| + |B| - 1$ lines are created and each edge is covered according to $f^{\min}$.

When entering the for-loop in line 23, all edges $\{v_i, v_0\}$ corresponding to elements $x_i \in S' := \bigcup_{k=1}^{K} (A^k \cup B^k)$ are covered according to $f^{\min}$ and at most $\sum_{i=1}^{K} (|A_i| + |B_i| - 1) = |S'| - K$ lines are created. In the for-loop, the remaining edges corresponding to $x_i \in S \setminus S'$ are covered by one one-edge line with appropriate frequency each. Thus, the corresponding line concept $(\mathcal{L}, f)$ is feasible.

The costs of the line concept correspond to the number of created line, and we get

$$\text{cost}((\mathcal{L}, f)) \le |S'| - K + |S \setminus S'| = |S| - K = m - K = K',$$

such that $(\mathcal{L}, f)$ is feasible for the decision version of LPAL.

For the other direction, consider a solution $(\mathcal{L}, f)$ to I with cost at most K', i.e. with at most $m - K$ lines. As discussed above, we may assume that one-edge lines do not overlap with other lines. We construct an auxiliary graph $H_{\mathcal{L}}$ with the vertices $v_1, \dots, v_m$. For each two-edge line (v_i, v_0, v_j), we create an edge $\{v_i, v_j\}$ in $H_{\mathcal{L}}$. As discussed above, we can adapt $(\mathcal{L}, f)$ such that $H_{\mathcal{L}}$ has no cycles. For each one-edge line (v_i, v_0), we delete the vertex v_i. Note that we do not need to delete edges since we assumed that one-edge lines do not overlap. Let m_1 be the number of one-edge lines, and m_2 the number of two-edge lines in $\mathcal{L}$. Now $H_{\mathcal{L}}$ has $m - m_1$ vertices and $m_2 \le K' - m_1 = m - m_1 - K$ edges.

As argued above, $H_{\mathcal{L}}$ is a forest and contains at least K components. We call these trees $T_1, \dots, T_K$. As we deleted all vertices belonging to one-edge lines, every vertex of $H_{\mathcal{L}}$ has at least one incident edge and thus each tree T_i contains at least two vertices.

Since every tree is bipartite, we may choose a bipartition $(P_1 := \{v_j : j \in J_1\}, P_2 := \{v_j : j \in J_2\})$ for every T_i. By construction, every line $\ell \in \mathcal{L}$ is either disjoint from $T_i = P_1 \cup P_2$ or connects P_1 with P_2, i.e. $\emptyset \ne V(\ell) \cap P_1$ if and only if $\emptyset \ne V(\ell) \cap P_2$. We obtain:

$$\sum_{j \in J_1} x_j = \sum_{j \in J_1} \sum_{\substack{\ell \in \mathcal{L}: \\ \{v_j, v_0\} \in E(\ell)}} f_\ell = \sum_{\substack{\ell \in \mathcal{L}: \\ \emptyset \ne V(\ell) \cap P_1}} f_\ell = \sum_{\substack{\ell \in \mathcal{L}: \\ \emptyset \ne V(\ell) \cap P_2}} f_\ell = \sum_{j \in J_2} \sum_{\substack{\ell \in \mathcal{L}: \\ \{v_j, v_0\} \in E(\ell)}} f_\ell = \sum_{j \in J_2} x_j$$

Algorithm 1 Constructing a line concept from a Partition into many Partitions solution.

Input: a solution Sol $= ((A_1, B_1), \dots, (A_K, B_K))$ to a Partition into many Partitions instance $S = \{x_1, \dots, x_m\}$

```
1: $\mathcal{L} = \emptyset$
2: for $(A, B) \in$ Sol do
3:     treat the numbers in $A$ and $B$ as mutable data structures, which can store a value
       and an index
4:     assign to each $y$ in $A$ and $B$ an index $i$ such that $x_i = y$
5:     while $|A| > 0$ and $|B| > 0$ do
6:         $a = \min(A)$
7:         $b = \min(B)$
8:         if $a < b$ then
9:             add to $\mathcal{L}$ a line from $v_{a.\text{index}}$ to $v_{b.\text{index}}$ with frequency $a$
10:            A.remove(a)
11:            b -= a
12:        else if $a > b$ then
13:            add to $\mathcal{L}$ a line from $v_{a.\text{index}}$ to $v_{b.\text{index}}$ with frequency $b$
14:            B.remove(b)
15:            a -= b
16:        else
17:            add to $\mathcal{L}$ a line from $v_{a.\text{index}}$ to $v_{b.\text{index}}$ with frequency $a$
18:            A.remove(a)
19:            B.remove(b)
20:        end if
21:    end while
22: end for
23: for $i \in [1, n]$ where $\{v_i, v_0\}$ is not covered yet do
24:     add to $\mathcal{L}$ a line from $v_i$ to $v_0$ with frequency $x_i$
25: end for
26: return $(\mathcal{L}, f)$
```

and, hence, $(\{x_j : j \in J_1\}, \{x_j : j \in J_2\})$ is a nonempty solution to Partition. We repeat this for every tree T_i to get K disjoint number partitions, solving Partition into many Partitions.

The hardness for the case $f^{\max} \equiv \infty$ follows with Lemma 6. ◀

The presented hardness results in this section actually show *strong* NP-hardness, i.e. even when we restrict the numerical parameters of LPAL instances to be (polynomially) small compared to the graph, the problem remains NP-hard.

5 Optimal line planning for stars

While LPAL is NP-hard for paths if $d_{\text{fix}} > 0$, the problem is easier when no frequency-independent costs are considered, i.e., for $d_{\text{fix}} = 0$. Here, the costs do not increase if edges are covered by multiple lines, ending at different terminals. We can show that optimal solutions have a special structure by rewriting the cost function

$$\text{cost}((\mathcal{L}, f)) = \underbrace{d_{\text{fix}}}_{=0} \cdot |\mathcal{L}| + \sum_{\ell \in \mathcal{L}} \text{cost}_\ell \cdot f_\ell = \sum_{e \in E} c_e \cdot F_e^{(\mathcal{L}, f)} + c_{\text{fix}} \cdot \sum_{\ell \in \mathcal{L}} f_\ell. \tag{1}$$

As all edges in a star are incident to a central vertex, there is an optimal solution where each edge $e \in E$ is covered exactly $f_e^{\min}$ times, i.e. $F_e^{(\mathcal{L},f)} = f_e^{\min}$. Thus, it remains only to minimize the frequency-dependent fixed costs $c_{\text{fix}} \cdot \sum_{\ell \in \mathcal{L}} f_\ell$ in (1). As each line contains either one or two edges and two-edge lines reduce the costs by c_{fix}, this is equivalent to minimizing the total frequency of one-edge lines.

It is easy to see that each of the following conditions guarantees optimality of the line concept as in each case as many edges as possible are "paired up" to two-edge lines:

1. There is no one-edge line.
2. There is one one-edge line with frequency one.
3. There is an edge with $\bar{e} \in E$ with $f_{\bar{e}}^{\min} > \sum_{e \in E \setminus \{\bar{e}\}} f_e^{\min}$ and $\sum_{\ell \in \mathcal{L}} f_\ell = f_{\bar{e}}^{\min}$.

Algorithm 2 Finding an optimal solution to LPAL for stars.

```
Input: An instance $(G, d_{\text{fix}}, c_{\text{fix}}, c, f^{\min}, f^{\max})$ where $d_{\text{fix}} = 0$ and $G = (V, E)$ is a star.
 1: $E_{\text{list}} = [e_1, \dots, e_m]$ list of edges in $E$, sorted decreasingly by $f_e^{\min}$, $\bar{E} = \emptyset$, $\bar{f}^{\min} = f^{\min}$
 2: $f_{(e)} = 0$, $f_{(e_i,e_j)} = 0$ for all $e, e_i, e_j \in E$, $i > j$
 3: for $e_k \in E_{\text{list}}$ do
 4:     if there is $\bar{e} \in \bar{E}$ then
 5:         $a = \min\{f_{(\bar{e})}, \bar{f}_{e_k}^{\min}\}$
 6:         $f_{(\bar{e})}$ −= $a$, $f_{(e_k,\bar{e})} = a$
 7:         $\bar{f}_{e_k}^{\min}$ −= $a$
 8:         if $f_{(\bar{e})} = 0$ then
 9:             $\bar{E} = \emptyset$
10:         end if
11:     end if
12:     for $e_i, e_j \in \{e_1, \dots, e_{k-1}\}$ with $i > j$, $f_{(e_i,e_j)} > 0$ and $\bar{f}_{e_k}^{\min} > 1$ do
13:         $b = \min\left\{ f_{(e_i,e_j)}, \left\lfloor \frac{\bar{f}_{e_k}^{\min}}{2} \right\rfloor \right\}$
14:         $f_{(e_i,e_j)}$ −= $b$, $f_{(e_k,e_i)}$ += $b$, $f_{(e_k,e_j)}$ += $b$
15:         $\bar{f}_{e_k}^{\min}$ −= $2 \cdot b$
16:     end for
17:     if $\bar{f}_{e_k}^{\min} > 0$ then
18:         $f_{(e_k)} = \bar{f}_{e_k}^{\min}$, $\bar{E} = \{e_k\}$
19:     end if
20: end for
21: $\mathcal{L} = \{(e_i, e_j) \colon f_{(e_i,e_j)} > 0\} \cup \{(e) \colon f_{(e)} > 0\}$, $f = f|_{\mathcal{L}}$
22: return $(\mathcal{L}, f)$
```

In Algorithm 2, we present a polynomial time algorithm that finds an optimal solution to LPAL. Starting with a list of edges sorted by decreasing $f_e^{\min}$, LPAL is iteratively solved for the first k edges, $k \in \{1, \dots, |E|\}$ such that one of the optimality conditions 1, 2 or 3 is satisfied at the end of each iteration for the already considered edges. The one-edge lines with positive frequency are stored in the set $\bar{E}$ which never contains more than one edge.

After iteration 1, $\bar{E} = \{e_1\}$ and condition 3 is satisfied. In iteration k, the edge e_k is paired up with edge $\bar{e} \in \bar{E}$ creating a new two-edge line if $\bar{E}$ is not empty. If $f_{(\bar{e})} > \bar{f}_{e_k}^{\min}$, $\bar{f}_{e_k}^{\min}$ is reduced to zero, $\bar{E} = \{\bar{e}\}$ and condition 3 is satisfied. If $f_{(\bar{e})} = \bar{f}_{e_k}^{\min}$, $\bar{f}_{e_k}^{\min}$ and $f_{(\bar{e})}$ are reduced to zero, $\bar{E} = \emptyset$ and condition 1 is satisfied. If $f_{(\bar{e})} < \bar{f}_{e_k}^{\min}$or $\bar{E} = \emptyset$ in line 4, $\bar{E} = \emptyset$ in the for-loop starting in line 12 and we have to show that at the end of the iteration either condition 1 or 2 is satisfied. As the list of edges is sorted by decreasing $f_e^{\min}$, we know that the total frequency of all already constructed lines is at least $\frac{f^{\min}_{e_k}}{2}$ such that we can split already existing lines and create two new ones containing e_k. Thus, in line 17 $\bar{f}^{\min}_{e_k}$ is either zero or one, such that optimality condition 1 or 2 is satisfied and we get the following theorem, proven in Appendix B.

▶ **Theorem 11.** *Algorithm 2 finds an optimal solution to* LPAL *for stars with* $d_{\mathit{fix}} = 0$ *in* $\mathcal{O}(n^3)$.

6 Optimal line planning for trees

Since paths are special instances of trees, LPAL is NP-hard on trees by Theorem 7. If we assume that $d_{\text{fix}} = 0$ and that $f^{\max}$ is bounded by a constant b, then we can provide a pseudo-linear time algorithm for finding the optimal objective value of LPAL on trees.

▶ **Theorem 12.** *If T is a tree, $d_{\mathit{fix}} = 0$, and $f^{\max}$ is bounded by a constant b, then the minimal cost for* LPAL *can be computed in $\mathcal{O}(nb^3)$. An optimal line concept can be computed in $\mathcal{O}(n^3b^3)$.*

Intuition. It is well known that a rooted tree (T, r) can be constructed from the set $S := \{((\{v\}, \emptyset), v) \colon v \text{ is a leaf of } T\}$ of rooted singleton trees by iteratively introducing parents and merging subtrees. For our dynamic program it is crucial that we restrict the operations further. We iteratively modify S by the following two operations:

- *introduce a parent:* extend (T', r') by $p \in V(T) \setminus V(T')$ if p is the only neighbor of r' in T that is not contained in $V(T')$. Replace (T', r') in S by the extended tree.
- *merge:* (T_1, r') and (T_2, r') can be merged at r' if r' has only one child in T_1 or in T_2. Replace the two trees in S by the merged tree.

If none of the above two operations can be applied, then $S = \{(T, r)\}$. We exploit that there exists an optimal solution for LPAL with the following property: the restriction of this solution to a rooted tree (T', r') arising in the above construction satisfies that at most b lines end in r' (otherwise a merge of two such lines would give a solution of lower costs). We compute the optimal value for LPAL using the above construction where each subtree has a table which stores its optimal solutions, considering any possible number of lines ending in its root. If $(\mathcal{L}, f)$ is a line concept for T, then for each $v \in V(T)$ we define *the number of lines ending at v* as

$$\eta_v((\mathcal{L}, f)) := \sum_{\substack{\ell \in \mathcal{L} \colon v \text{ is} \\ \text{an end of } \ell}} f_\ell$$

where we allow zero-edge lines. The cost of an optimal solution satisfying $\eta_v \geq k$ is

$$\operatorname{cost}(T \mid \eta_v \geq k) := \min\{\operatorname{cost}((\mathcal{L}, f)) \mid (\mathcal{L}, f) \in \mathcal{F}(T),\ \eta_v((\mathcal{L}, f)) \geq k\}.$$

We compute the *cost vector*

$$\operatorname{cost}(T', r') := (\operatorname{cost}(T' \mid \eta_{r'} \geq 0), \operatorname{cost}(T' \mid \eta_{r'} \geq 1), \ldots, \operatorname{cost}(T' \mid \eta_{r'} \geq b))$$

for each rooted subtree (T', r') appearing in the above construction. The recursive computation stores intermediate results in a table to avoid re-computation. Finally, the cost of an optimal line concept for T is $\operatorname{cost}(T \mid \eta_r \geq 0)$.

▶ **Lemma 13** (Dynamic programming for trees). *Let (T', r') be a rooted tree and $k \in \{1, \dots, b\}$.*

1. Singletons: *If $|V(T')| = 1$, then $\operatorname{cost}(T' \mid \eta_{r'} \geq k) = k \cdot c_{\mathit{fix}}$. The time required to compute $\operatorname{cost}(T' \mid \eta_{r'} \geq k)$ is $\mathcal{O}(1)$ and, hence, the time required to compute $\operatorname{cost}(T', r')$ is $\mathcal{O}(b)$.*
2. Introduce a parent: *If $\deg_{T'}(r') = 1$ and u denotes the child of r' in T', then $\operatorname{cost}(T' \mid \eta_{r'} \geq k)$ equals*

$$\min_{0 \leq m \leq \max\{k, f^{\min}_{\{u,r'\}}\}} \{\operatorname{cost}(T' - r' \mid \eta_u \geq m) + \max\left\{k, f^{\min}_{\{u,r'\}}\right\} \cdot (c_{\mathit{fix}} + c_{\{u,r'\}}) - m \cdot c_{\mathit{fix}}\}.$$

   If the values $\operatorname{cost}(T' - r' \mid \eta_u \geq m)$ are pre-computed for all $m \in \{1, \dots, b\}$, then the time required to compute $\operatorname{cost}(T' \mid \eta_{r'} \geq k)$ is $\mathcal{O}(b)$ and, hence, $\operatorname{cost}(T', r')$ can be computed in $\mathcal{O}(b^2)$ time.
3. Merge: *If (T', r') is the union of two rooted trees $(T_1, r'), (T_2, r')$ where $\deg_{T_1}(r') = 1$, then*

$$\operatorname{cost}(T' \mid \eta_{r'} \geq k) = \min_{\substack{0 \leq m, k_1, k_2 \leq b, \\ k_1 + k_2 - 2m = k}} \{\operatorname{cost}(T_1 \mid \eta_{r'} \geq k_1) + \operatorname{cost}(T_2 \mid \eta_{r'} \geq k_2) - m \cdot c_{\mathit{fix}}\}.$$

   If the values $\operatorname{cost}(T_2 \mid \eta_{r'} \geq k_2)$ and $\operatorname{cost}(T_2 \mid \eta_{r'} \geq k_2)$ are pre-computed for all $k_1, k_2 \in \{1, \dots, b\}$, then the time required to compute $\operatorname{cost}(T' \mid \eta_{r'} \geq k)$ is $\mathcal{O}(b^2)$ and, hence, it requires $\mathcal{O}(b^3)$ time to compute $\operatorname{cost}(T', r')$.

   For a proof of Lemma 13, see Appendix C.

Total runtime. A depth-first search algorithm yields a decomposition of T such that the dynamic programming approach can be executed in the corresponding order. Since T is a tree DFS has a running time of $\mathcal{O}(n)$. The running time to compute the cost vector for all leaves in the initial set S is in $\mathcal{O}(nb)$ since there are at most $n - 1$ leaves in T and by Lemma 13.(1). In the construction of T we introduce a parent $|E(T)| = n - 1$ times. Together with Lemma 13.(2) this yields that computing the respective cost vectors has a total running time of $\mathcal{O}(nb^2)$. The merge operation is performed $\mathcal{O}(\sum_{v \in V(T)} \deg_T(v)) = \mathcal{O}(n)$ times which gives a total running time of $\mathcal{O}(nb^3)$. Altogether, the dynamic programming has a running time of $\mathcal{O}(nb^3)$.

Constructing a line concept. We showed how to compute the minimal cost among all feasible line concepts. To *construct* a line concept of that cost we store in each cost vector entry additionally a line concept of that cost. These line concepts can be computed recursively, according to the decisions made (i.e. creating zero-edge line, extending lines by a single edge, joining lines). This increases the algorithm runtime, depending on the line concept sizes. On a tree there are $\mathcal{O}(n^2)$ different paths. It is then possible to compute all cost vectors augmented with line concepts in time $\mathcal{O}(n^3 b^3)$. Altogether, this proves Theorem 12.

Since the runtime of the algorithm depends on b, it is *pseudo-polynomial.* For the special case where for all $e \in E$ it holds $f^{\min}_e = f^{\max}_e$, we provide a true polynomial time algorithm, which does not depend on a frequency bound b.

▶ **Theorem 14.** *If G is a tree, $d_{\mathit{fix}} = 0$, and $f^{\min}_e = f^{\max}_e$ for all $e \in E$, then Algorithm 3 computes an optimal solution to* LPAL *in $\mathcal{O}(n^3)$.*

The key idea of Algorithm 3 is to apply Algorithm 2 iteratively at every vertex. As $f^{\min} = f^{\max}$, we can handle lines ending at vertex $v \in V$ in the same way we handle edges in stars: creating a two-edge line in a star corresponds to concatenating two lines in a tree. A formal proof can be found in Appendix D.

Algorithm 3 Finding an optimal solution of LPAL on trees with $f^{\min} = f^{\max}$.

Input: An instance $(G, d_{\text{fix}}, c_{\text{fix}}, c, f^{\min}, f^{\max})$ where $d_{\text{fix}} = 0$, $f^{\min} = f^{\max}$ and $G = (V, E)$ is a tree.

```
1:  $\mathcal{L} = \{(e) : e \in E\}$
2:  $f_{(e)} = f_e^{\min}$ for all $e \in E$; for all other paths $\ell$ set $f_\ell = 0$
3:  for $v \in V$ do
4:      let $S$ be the star formed by $v$ and its neighbors
5:      let $(\mathcal{L}^S, f^S)$ be the result of Algorithm 2 applied to the sub-instance on $S$
6:      $L_v = \{\ell \in \mathcal{L} : \ell$ ends in $v\}$
7:      for $\ell_1, \ell_2 \in L_v$ do
8:          let $e_1$ be the edge of $\ell_1$ incident to $v$
9:          let $e_2$ be the edge of $\ell_2$ incident to $v$
10:         if $e_1 = e_2$ then
11:             continue
12:         end if
13:         $d = \min\{f^S_{(e_1,e_2)}, f_{\ell_1}, f_{\ell_2}\}$
14:         $\ell_+ = \ell_1 \cup \ell_2$
15:         $\mathcal{L} = \mathcal{L} \cup \{\ell_+\}$
16:         $f^S_{(e_1,e_2)}$ $-{=}$ $d$, $f_{\ell_1}$ $-{=}$ $d$, $f_{\ell_2}$ $-{=}$ $d$, $f_{\ell_+}$ $+{=}$ $d$
17:     end for
18: end for
19: $\mathcal{L} = \{\ell \in \mathcal{L} : f_\ell > 0\}$, $f = f|_{\mathcal{L}}$
20: return $(\mathcal{L}, f)$
```

7 Conclusion and outlook

Line planning on all lines LPAL means allowing all simple paths as possible lines in a public transport supply. This large search space yields more options and, hence, better solutions for optimal public transport planning. In this paper, we illuminated the algorithmic aspects of LPAL. Frequency-independent line costs result in an NP-hard problem even for paths and stars. Without these costs LPAL remains NP-hard on planar graphs but can be solved in polynomial time on trees when $f^{\min} \equiv f^{\max}$, and in pseudo-polynomial time otherwise. Assuming $P \neq NP$, no useful approximation algorithm can exist, unless we further restrict the problem inputs. Even when $f^{\max} \equiv \infty$, no constant-factor approximation is possible. The following are the most pressing open questions:

- Is LPAL in NP? It is not clear that, especially when $f^{\min}$ is very large, the size of an optimal line concept can be bounded by a polynomial in the input size.
- Is there a polynomial time algorithm for LPAL with $d_{\text{fix}} = 0$ on trees?
- Is there a (pseudo-)polynomial time algorithm for LPAL with $d_{\text{fix}} = 0$ on graphs with treewidth 2 (or generally bounded treewidth)?
- Under which restrictions exists a constant-factor approximation algorithm for LPAL?

When moving from trees to graphs of higher treewidth, an additional degree of freedom can be considered: while for trees we can assume that passenger paths are fixed, this is no longer true in general graphs. Thus, replacing the lower frequency bounds $f^{\min}$ by a flow formulation for the passengers as in [3] can lead to even better solutions from a passenger's point of view. This presents an interesting extension of the problem, where it is especially important to understand the structure of optimal solutions.

References

1 R. Arbex and C. da Cunha. Efficient transit network design and frequencies setting multi-objective optimization by alternating objective genetic algorithm. *Transportation Research Part B: Methodological*, 81:355–376, 2015. doi:10.1016/j.trb.2015.06.014.

2 R. Borndörfer, O. Arslan, Z. Elijazyfer, H. Güler, M. Renken, G. Şahin, and T. Schlechte. Line planning on path networks with application to the istanbul metrobüs. In *Operations Research Proceedings 2016*, pages 235–241. Springer, 2018. doi:10.1007/978-3-319-55702-1_32.

3 R. Borndörfer, M. Grötschel, and M. Pfetsch. A column-generation approach to line planning in public transport. *Transportation Science*, 41(1):123–132, 2007. doi:10.1287/trsc.1060.0161.

4 S. Bull, J. Larsen, R. Lusby, and N. Rezanova. Optimising the travel time of a line plan. *4OR*, October 2018. doi:10.1007/s10288-018-0391-5.

5 M. Bussieck, P. Kreuzer, and U. Zimmermann. Optimal lines for railway systems. *European Journal of Operational Research*, 96(1):54–63, 1997. doi:10.1016/0377-2217(95)00367-3.

6 H. Cancela, A. Mauttone, and M. E. Urquhart. Mathematical programming formulations for transit network design. *Transportation Research Part B: Methodological*, 77:17–37, 2015. doi:10.1016/j.trb.2015.03.006.

7 M. Claessens, N. van Dijk, and P. Zwaneveld. Cost optimal allocation of rail passenger lines. *European Journal of Operational Research*, 110(3):474–489, 1998. doi:10.1016/S0377-2217(97)00271-3.

8 C. J. Colbourn. The complexity of completing partial latin squares. *Discrete Applied Mathematics*, 8(1):25–30, 1984. doi:10.1016/0166-218X(84)90075-1.

9 R. Z. Farahani, E. Miandoabchi, W. Y. Szeto, and H. Rashidi. A review of urban transportation network design problems. *European Journal of Operational Research*, 229(2):281–302, 2013. doi:10.1016/j.ejor.2013.01.001.

10 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

11 P. Gattermann. Generating Line-Pools. Master's thesis, Fakultät für Mathematik und Informatik, Georg-August-University Göttingen, 2015.

12 P. Gattermann, J. Harbering, and A. Schöbel. Line pool generation. *Public Transport*, 9(1-2):7–32, 2017. doi:10.1007/s12469-016-0127-x.

13 M. Goerigk and M. Schmidt. Line planning with user-optimal route choice. *European Journal of Operational Research*, 259(2):424–436, 2017. doi:10.1016/j.ejor.2016.10.034.

14 V. Guihaire and J. Hao. Transit network design and scheduling: A global review. *Transportation Research Part A: Policy and Practice*, 42(10):1251–1273, 2008. doi:10.1016/j.tra.2008.03.011.

15 H. Hulett, T. G. Will, and G. J. Woeginger. Multigraph realizations of degree sequences: Maximization is easy, minimization is hard. *Operations Research Letters*, 36(5):594–596, 2008. doi:10.1016/j.orl.2008.05.004.

16 R. M. Karp. Reducibility Among Combinatorial Problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

17 K. Kepaptsoglou and M. Karlaftis. Transit route network design problem. *Journal of transportation engineering*, 135(8):491–505, 2009. doi:10.1061/(ASCE)0733-947X(2009)135:8(491).

18 B. Masing, N. Lindner, and R. Borndörfer. The Price of Symmetric Line Plans in the Parametric City. *arXiv preprint arXiv:2201.09756*, 2022. doi:10.48550/ARXIV.2201.09756.

19 J. Pätzold, A. Schiewe, and A. Schöbel. Cost-Minimal Public Transport Planning. In R. Borndörfer and S. Storandt, editors, *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*, volume 65 of *OpenAccess*

Series in Informatics (OASIcs), pages 8:1–8:22. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018. doi:10.4230/OASIcs.ATMOS.2018.8.

20 J. Plesník. The NP-completeness of the Hamiltonian cycle problem in planar digraphs with degree bound two. *Information Processing Letters*, 8(4):199–201, 1979. doi:10.1016/0020-0190(79)90023-1.

21 A. Schiewe, P. Schiewe, and M. Schmidt. The line planning routing game. *European Journal of Operational Research*, 274(2):560–573, 2019. doi:10.1016/j.ejor.2018.10.023.

22 A. Schöbel. Line planning in public transportation: models and methods. *OR spectrum*, 34(3):491–510, 2012. doi:10.1007/s00291-011-0251-6.

23 A. Schöbel and S. Scholl. Line planning with minimal transfers. In *5th Workshop on Algorithmic Methods and Models for Optimization of Railways*, number 06901 in Dagstuhl Seminar Proceedings, 2006. doi:10.4230/OASIcs.ATMOS.2005.660.

24 L. M. Torres, R. Torres, R. Borndörfer, and M. E. Pfetsch. Line Planning on Paths and Tree Networks with Applications to the Quito Trolebu´s System. In Matteo Fischetti and Peter Widmayer, editors, *8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'08)*, volume 9 of *OpenAccess Series in Informatics (OASIcs)*, Dagstuhl, Germany, 2008. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/OASIcs.ATMOS.2008.1583.

25 L. M. Torres, R. Torres, R. Borndörfer, and M. E. Pfetsch. Line planning on tree networks with applications to the Quito Trolebús system. *International Transactions in Operational Research*, 18(4):455–472, 2011. doi:10.1111/j.1475-3995.2010.00802.x.

26 Q. K. Wan and H. K. Lo. A mixed integer formulation for multiple-route transit network design. *J. Math. Model. Algorithms*, 2(4):299–308, 2003. doi:10.1023/B:JMMA.0000020425.99217.cd.

27 G. Şahin, A. Ahmadi Digehsara, R. Borndörfer, and T. Schlechte. Multi-period line planning with resource transfers. *Transportation Research Part C: Emerging Technologies*, 119:102726, 2020. doi:10.1016/j.trc.2020.102726.

A Proof of Lemma 10

▶ **Lemma 10.** *Partition into many Partitions is strongly NP-hard.*

Proof. This proof is an adaptation of the reduction described in [15]. We reduce Partial Latin Square Completion (cf. [15]) to Partition into many Partitions.

Consider a partial Latin square L of dimension $p \times p$ with m missing entries. Define $q := 6p - 2$. We construct a Partition into many Partitions instance from the Latin square by defining $K := m$ and putting the following numbers into the set S:

- If color c does not occur in row k, put $x(k, c) := q(2k - 1) - (2c - 1)$ into S.
- If color c does not occur in column ℓ, put $y(\ell, c) := q^2(2\ell - 1) + (2c - 1)$ into S.
- If the cell in row k and column ℓ is empty, put $z(k, \ell) := q^2(2\ell - 1) + q(2k - 1)$ into S.

Partition into many Partitions requires that S only contains positive numbers. A quick check of the x-numbers shows that they are positive: $x(k, c) \geq q - (2c - 1) \geq q - (2p - 1) = 4p - 1 > 0$. The y- and z-numbers are positive since ℓ, c and k each are positive.

We check that these numbers indeed form a set of size $3m$, i.e. they are pairwise different: Assume $x(k_1, c_1) = x(k_2, c_2)$ holds for some $k_1, c_1, k_2, c_2 \in [1, p]$. Considering this equation modulo q, we find $(2c_1 - 1) \equiv (2c_2 - 1) \mod q$. Since $q > 2p - 1$, it follows: $c_1 = c_2$. Hence the equation simplifies to $q(2k_1 - 1) = q(2k_2 - 1)$, so also $k_1 = k_2$. This shows that the x-numbers are created by an injective map. The same arguments work for pairs of y-numbers and pairs of z-numbers. Now assume $x(k_1, c_1) = y(\ell_2, c_2)$. It follows that $(2c_1 - 1) + (2c_2 - 1) \equiv 0 \mod q$. This is a contradiction since $4p - 2 < q$. Assume $x(k_1, c_1) = z(k_2, \ell_2)$ or $y(\ell_1, c_1) = z(k_2, \ell_2)$. In both cases $(2c_1 - 1) \equiv 0 \mod q$ would follow, which is a contradiction.

Now we consider all the ways 3 or fewer of these numbers can be a yes-instance to PARTITION. A single number cannot be a yes-instance. Two numbers also cannot be a yes-instance, since S is a set and all numbers are pairwise distinct. Here, we work out only some of the possible three-number combinations. The rest can be calculated similarly.

- $z(k_1, \ell_1) = x(k_2, c_2) + y(\ell_3, c_3)$. Considering this equation modulo q, we find that $c_2 = c_3$. Then, dividing by q and again applying modulo, we get $k_1 = k_2$ and finally $\ell_1 = \ell_3$.
- $z(k_1, \ell_1) + x(k_2, c_2) = y(\ell_3, c_3)$. It would follow: $(2c_2 - 1) + (2c_3 - 1) \equiv 0 \mod q$, which is not possible, as we have seen before.
- $x(k_1, c_1) = y(\ell_2, c_2) + y(\ell_3, c_3)$. It would follow: $(2c_1 - 1) + (2c_2 - 1) + (2c_3 - 1) \equiv 0 \mod q$. This is not possible, since $0 < (2c_1 - 1) + (2c_2 - 1) + (2c_3 - 1) \leq 6p - 3 < q$.
- $x(k_1, c_1) = x(k_2, c_2) + x(k_3, c_3)$. Consider this equation modulo 2. Since q is even, we would obtain $-1 \equiv -2 \mod 2$, which is a contradiction. The case of three y-numbers is dealt with in the same way. In the case of three z-numbers, first divide by q.

After considering all combinations, we find that the only way three numbers can be a yes-instance to PARTITION, is by choosing one number from each family x,y and z; importantly these numbers have matching choices for row, column and color.

Now let $B_1, \dots, B_m$ be a solution to PARTITION INTO MANY PARTITIONS, i.e. the B_i are nonempty yes-instances to PARTITION, are pairwise disjoint and their union is a subset of S. As we have shown, each B_i contains at least three elements. Since $|S| = 3m$, every element of S is used and no B_i can contain more than three elements. Then each B_i corresponds to a triple of x-,y- and z-numbers, which in turn corresponds to a row k, a column ℓ and a color c. We then fill our partial Latin square, by coloring the cell at row k and column ℓ with c, repeating this for every B_i. Since every z-number was used, the Latin square is filled. It is also a valid coloring, since for every row/column each missing color appears only in one x-number/y-number.

For the other direction, consider a valid completion of the partial Latin square. Then for each of the m new colorings c_i in the cell at row k_i and column ℓ_i, we create $B_i := \{z(k_i, \ell_i), x(k_i, c_i), y(\ell_i, c_i)\}$. Then each B_i is a yes-instance to PARTITION, and is contained in S. The created sets are pairwise disjoint, since the Latin square would otherwise have a collision.

This reduction proves *strong* NP-hardness: we may assume without loss of generality that the input Latin square has at least p missing entries, because otherwise it would have a completely filled row, from which we could remove an arbitrary cell without affecting its ability to be completed. Then $|S| \geq 3p$ and the numbers in S are bounded by a polynomial in p, hence also by a polynomial in $|S|$. ◀

B Proof of Theorem 11

▶ **Theorem 11.** *Algorithm 2 finds an optimal solution to* LPAL *for stars with* $d_{\text{fix}} = 0$ *in* $\mathcal{O}(n^3)$.

Proof. Note that Algorithm 2 computes a line concept that covers each edge $e \in E$ exactly $f_e^{\min}$ times, i.e. $F_e^{(\mathcal{L},f)} = f_e^{\min}$. To prove optimality for $d_{\text{fix}} = 0$, we therefore only have to show that the total frequency of one-edge lines is minimized.

At the start of each for loop in line 3, the set $\bar{E}$ contains the edges for which a one-edge line with positive frequency exists. Note that there is always at most one edge $\bar{e} \in \bar{E}$, as by the choice of a in line 5, $f_e^{\min}$ can only be positive if $f_{(\bar{e})}$ is set to zero. Thus the line concept $(\mathcal{L}, f)$ created in line 22 contains at most one one-edge line with positive frequency.

If there is no one-edge line, the line concept is optimal as in condition 1.

If there is a one-edge line containing the first edge e_1 of E_{list}, i.e. the edge with the highest $f^{\min}$, then in line 5 the minimum a is always chosen as $f_e^{\min}$ for $e \neq e_1$, i.e. $f_{e_1}^{\min} > \sum_{e \neq e_1} f_e^{\min}$. In this case, all lines contain edge e_1 and thus $\sum_{\ell \in \mathcal{L}} f_\ell = f_{e_1}^{\min}$ such that $(\mathcal{L}, f)$ is optimal, see condition 3.

Otherwise, there is a one-edge line that does not contain the first edge. Here, we show that for every $e_k \neq e_1$ in the outer-loop (lines 3 to 20) with $\bar{f}_{e_k}^{\min} > 0$ in line 17 also $\bar{f}_{e_k}^{\min} = 1$ holds. Then, we have one one-edge line with frequency one and the line concept is optimal according to condition 2.

As $\bar{f}_{e_k}^{\min}$ is only reduced in the algorithm, $\bar{f}_{e_k}^{\min} > 0$ can only hold in line 17 if it already holds before the for-loop starting in line 12. Note that in this case, $k > 2$ holds. We want to show that $\bar{f}_{e_k}^{\min}$ is reduced in the for-loop (lines 12 to 16) until $\bar{f}_{e_k}^{\min} \in \{0, 1\}$. Suppose to the contrary, that $\bar{f}_{e_k}^{\min} > 1$ in line 17. Then the minimum b chosen in line 13 always has been chosen as $f_{(e_i,e_j)}$ and we get

$$\sum_{\substack{(e_i,e_j):\\ i,j<k}} f_{(e_i,e_j)} < \alpha$$

where α is the value of $\bar{f}_{e_k}^{\min}$ before starting the for-loop in line 12. We know that $\alpha = f_{e_k}^{\min}$ if $\bar{E} = \emptyset$ in line 4 and $\alpha = f_{e_k}^{\min} - f_{(e_k,\bar{e})}$ if $\bar{E} = \{\bar{e}\}$ in line 4. To simplify the notation in the following we set $f_{(e_k,\bar{e})} = 0$ if $\bar{E} = \emptyset$. As at the beginning of the for-loop in line 4 for e_k all edges e_i, $i \in \{1, \dots, k-1\}$, are covered $f_{e_i}^{\min}$-times we get

$$\sum_{\substack{(e_i,e_j):\\ i,j<k}} f_{(e_i,e_j)} + f_{(e_k,\bar{e})} \geq \frac{1}{2} \sum_{i=1}^{k-1} f_{e_i}^{\min} \geq \frac{1}{2} \cdot 2 \cdot f_{e_k}^{\min} = f_{e_k}^{\min}$$

and thus

$$\sum_{\substack{(e_i,e_j):\\ i,j<k}} f_{(e_i,e_j)} \geq f_{e_k}^{\min} - f_{(e_k,\bar{e})} = \alpha$$

which is the desired contradiction.

The runtime of Algorithm 2 can be estimated in the following way: there are $|E| = |V| - 1 = n - 1$ iterations of the outer for-loop starting in line 3 and $\mathcal{O}(n^2)$ iterations of the inner for-loop starting in line 12. As sorting E_{list} in line 1, initializing the frequencies in line 2 and reconstructing the line concept in line 22 are also in $\mathcal{O}(n^3)$, the total runtime of Algorithm 2 is $\mathcal{O}(n^3)$. ◀

C Proof of Lemma 13

▶ **Lemma 13** (Dynamic programming for trees). *Let (T', r') be a rooted tree and $k \in \{1, \dots, b\}$.*

1. Singletons: *If $|V(T')| = 1$, then $\text{cost}(T' \mid \eta_{r'} \geq k) = k \cdot c_{\mathit{fix}}$. The time required to compute $\text{cost}(T' \mid \eta_{r'} \geq k)$ is $\mathcal{O}(1)$ and, hence, the time required to compute $\text{cost}(T', r')$ is $\mathcal{O}(b)$.*
2. Introduce a parent: *If $\deg_{T'}(r') = 1$ and u denotes the child of r' in T', then $\text{cost}(T' \mid \eta_{r'} \geq k)$ equals*

$$\min_{0 \leq m \leq \max\{k, f_{\{u,r'\}}^{\min}\}} \{\text{cost}(T' - r' \mid \eta_u \geq m) + \max\left\{k, f_{\{u,r'\}}^{\min}\right\} \cdot (c_{\mathit{fix}} + c_{\{u,r'\}}) - m \cdot c_{\mathit{fix}}\}.$$

If the values $\text{cost}(T' - r' \mid \eta_u \geq m)$ are pre-computed for all $m \in \{1, \dots, b\}$, then the time required to compute $\text{cost}(T' \mid \eta_{r'} \geq k)$ is $\mathcal{O}(b)$ and, hence, $\text{cost}(T', r')$ can be computed in $\mathcal{O}(b^2)$ time.

3. Merge: *If (T', r') is the union of two rooted trees $(T_1, r'), (T_2, r')$ where $\deg_{T_1}(r') = 1$, then*

$$\operatorname{cost}(T' \mid \eta_{r'} \geq k) = \min_{\substack{0 \leq m, k_1, k_2 \leq b, \\ k_1 + k_2 - 2m = k}} \{\operatorname{cost}(T_1 \mid \eta_{r'} \geq k_1) + \operatorname{cost}(T_2 \mid \eta_{r'} \geq k_2) - m \cdot c_{\mathit{fix}}\}.$$

If the values $\operatorname{cost}(T_2 \mid \eta_{r'} \geq k_2)$ and $\operatorname{cost}(T_2 \mid \eta_{r'} \geq k_2)$ are pre-computed for all $k_1, k_2 \in \{1, \ldots, b\}$, then the time required to compute $\operatorname{cost}(T' \mid \eta_{r'} \geq k)$ is $\mathcal{O}(b^2)$ and, hence, it requires $\mathcal{O}(b^3)$ time to compute $\operatorname{cost}(T', r')$.

Proof. If (T', r') has only one vertex, then clearly the optimal line concept which satisfies that k lines end in r' consists of k zero-edge lines. This implies (1).

We prove (2). Since $\deg_{T'}(r') = 1$ every line ℓ in T' is either contained in $T' - r'$ or it has one end in $T' - r'$ and the other end is r'. In a line concept $(\mathcal{L}, f)$ of T', a line ℓ with one end in $T' - r'$, the other end being r' and frequency f_ℓ can be split into two lines $\ell_1 = (r', u)$ and $\ell_2 = \ell - r'$ with frequency f_ℓ without changing the feasibility. The line ℓ_2 is contained in $T' - r'$ and the cost of the line concept is increased by $c_{\text{fix}} \cdot f_\ell$. This process can be reversed, merging some line from $T' - r'$ that ends at u with the line (u, r'), decreasing the cost accordingly. Assuming $k \leq f^{\max}_{\{u,r'\}}$, this allows us to rewrite $\operatorname{cost}(T' \mid \eta_{r'} \geq k)$:

$$\begin{aligned}
&\operatorname{cost}(T' \mid \eta_{r'} \geq k) = \min\{\operatorname{cost}((\mathcal{L}, f))\colon (\mathcal{L}, f) \in \mathcal{F}(T'),\ \eta_{r'}((\mathcal{L}, f)) \geq k\} \\
&\overset{(a)}{=} \min\{\operatorname{cost}((\mathcal{L}', f')) + a \cdot (c_{\text{fix}} + c_{\{u,r'\}}) - m \cdot c_{\text{fix}}\colon (\mathcal{L}', f') \in \mathcal{F}(T' - r'), \\
&\qquad\qquad f^{\min}_{\{u,r'\}} \leq a \leq f^{\max}_{\{u,r'\}}, m \leq \eta_u((\mathcal{L}', f')), m \leq a, a \geq k\} \\
&\overset{(b)}{=} \min\{\operatorname{cost}((\mathcal{L}', f')) + \max\{k, f^{\min}_{\{u,r'\}}\} \cdot (c_{\text{fix}} + c_{\{u,r'\}}) - m \cdot c_{\text{fix}}\colon (\mathcal{L}', f') \in \mathcal{F}(T' - r'), \\
&\qquad\qquad \max\{k, f^{\min}_{\{u,r'\}}\} \leq f^{\max}_{\{u,r'\}}, m \leq \eta_u((\mathcal{L}', f')), m \leq \max\{k, f^{\min}_{\{u,r'\}}\}\} \\
&\overset{(c)}{=} \min_{0 \leq m \leq \max\{k, f^{\min}_{\{u,r'\}}\}} \{\operatorname{cost}(T' - r' \mid \eta_u \geq m) + \max\{k, f^{\min}_{\{u,r'\}}\} \cdot (c_{\text{fix}} + c_{\{u,r'\}}) - m c_{\text{fix}}\}
\end{aligned}$$

(a) We split the lines in T' into some set of lines $\mathcal{L}'$ on $T' - r'$, and a copies of the line (u, r') of which m are merged with lines from $\mathcal{L}'$. Then the number of ends at r' is exactly a and, hence, $a \geq k$. Furthermore $a \in [f^{\min}_{\{u,r'\}}, f^{\max}_{\{u,r'\}}]$. Each merge reduces the cost by c_{fix}.

(b) To minimize the cost, we have to minimize a: the only benefit of increasing a is that m can be increased but the factor of a outweighs m. Hence we replace a by its minimum possible value $\max\{k, f^{\min}_e\}$.

(c) Since $m \leq \eta_u((\mathcal{L}', f'))$ we can replace $\operatorname{cost}((\mathcal{L}', f'))$ by $\operatorname{cost}(T' - r' \mid \eta_u \geq m)$. The condition $\max\{k, f^{\min}_{\{u,r'\}}\} \leq f^{\max}_{\{u,r'\}}$ is fulfilled by the assumption on k. The remaining constraints are written as a subscript.

If $k > f^{\max}_{\{u,r'\}}$, then $\operatorname{cost}(T' \mid \eta_{r'} \geq k) = \infty$ since no feasible line concept with $\eta_{r'} \geq k$ exists.

The time to compute $\operatorname{cost}(T' \mid \eta_{r'} \geq k)$ for some k is $\mathcal{O}(b)$, since $\max\{k, f^{\min}_{\{u,r'\}}\} \leq b$. Hence $\operatorname{cost}(T', r')$ can be computed in $\mathcal{O}(b^2)$.

Finally, we prove (3). Every line in T' that traverses r' can be split into two lines, one contained in T_1 and the other contained in T_2. In reverse, we can join lines from different subtrees together at r'. Then

$$\operatorname{cost}(T' \mid \eta_{r'} \geq k) = \min_{\substack{0 \leq m, k_1, k_2 \leq b, \\ k_1 + k_2 - 2m = k}} \{\operatorname{cost}(T_1 \mid \eta_{r'} \geq k_1) + \operatorname{cost}(T_2 \mid \eta_{r'} \geq k_2) - m \cdot c_{\text{fix}}\}$$

Note that at most b lines of T_1 end at r' by the degree condition. The time required to compute $\text{cost}(T' \mid \eta_{r'} \geq k)$ for some k is $\mathcal{O}(b^2)$ since we have two degrees of freedom in the minimum expression. Hence $\text{cost}(T', r')$ can be computed in $\mathcal{O}(b^3)$. ◀

D Proof of Theorem 14

▶ **Theorem 14.** *If G is a tree, $d_{fix} = 0$, and $f_e^{\min} = f_e^{\max}$ for all $e \in E$, then Algorithm 3 computes an optimal solution to* LPAL *in $\mathcal{O}(n^3)$.*

Proof. We first show that Algorithm 3 computes an optimal feasible solution: after line 2, a feasible line concept is constructed. The operations in line 16 merge lines and, hence, the feasibility of $(\mathcal{L}, f)$ remains.

For showing optimality, we note that since the total frequencies $F_e^{(\mathcal{L},f)}$ are fixed for every $e \in E$, obtaining an optimal line concept $(\mathcal{L}, f)$ is equivalent to minimizing $\sum_{\ell \in \mathcal{L}} f_\ell$. Since every line has two ends, another equivalent quantity to minimize is the total number of line ends, weighted by f, i.e. $2 \sum_{\ell \in \mathcal{L}} f_\ell$.

Define $L_{v,e} := \{\ell \in \mathcal{L} \colon \ell \text{ ends in } v \text{ and traverses } e\}$. We need an invariant (I1) that holds before every iteration of the outer for-loop: for every vertex $v \in V$ that has not yet been chosen in the outer for-loop, we have $f_e^{\min} = \sum_{\ell \in L_{v,e}} f_\ell$. Clearly (I1) holds directly after executing line 2. The operations during an iteration only affect the local line ends, i.e. the number of ends at yet unvisited vertices is unchanged. Hence (I1) is maintained.

Another invariant (I2), that holds before every iteration of the inner loop, for every edge e incident to v, is $\sum_{\ell \in L_{v,e}} f_\ell = f_{(e)}^S + \sum_{e' \neq e} f_{(e,e')}^S$. It holds initially, since Algorithm 2 produces a feasible line concept, and we have $f_e^{\min} = f_{(e)}^S + \sum_{e' \neq e} f_{(e,e')}^S$; combine this with (I1) to obtain (I2). Let e_1 and e_2 be chosen during an iteration, after line 9. The operations inside the loop only affect lines that contain e_1 or e_2, hence for every $e \notin \{e_1, e_2\}$ (I2) is maintained. (I2) is also maintained for e_1, since $f_{(e_1,e_2)}^S$ and f_{ℓ_1} are changed by equal amounts. The same holds true for e_2.

We claim that after the inner for-loop finishes, we have $f_\ell^S = 0$ for all two-edge lines $\ell = (e_1, e_2)$ of $\mathcal{L}^S$. Suppose towards a contradiction that $f_{(e_1,e_2)}^S > 0$ for some $e_1 \neq e_2$. By (I2), $\sum_{\ell \in L_{v,e_1}} f_\ell = f_{(e_1)}^S + \sum_{e' \neq e_1} f_{(e_1,e')}^S > 0$, and similarly $\sum_{\ell \in L_{v,e_2}} f_\ell > 0$. Hence there exist $\ell_1 \in L_{v,e_1}$ and $\ell_2 \in L_{v,e_2}$ with $f_{\ell_1} > 0$ and $f_{\ell_2} > 0$. This is a contradiction since the inner for-loop would have chosen ℓ_1 and ℓ_2 at some point, after which $\min\{f_{\ell_1}, f_{\ell_2}, f_{(e_1,e_2)}^S\} = 0$.

Using (I2) again, we have $\sum_{\ell \in L_{v,e}} f_\ell = f_{(e)}^S$ after the inner for-loop. This means that we have $\sum_{e \text{ incident to } v} f_{(e)}^S =: x_v$ line ends, with multiplicity, at vertex v. The algorithm's locality implies that this number does not change in further iterations of the outer loop.

In total Algorithm 3 produces a line concept with $\sum_{v \in V} x_v$ line ends. Suppose that there is a better solution, i.e. a feasible line concept $(\mathcal{L}', f')$ that has fewer than x_v line ends at some vertex v. Then we could restrict $(\mathcal{L}', f')$ onto the star S around v and would obtain a solution for S which has fewer ends, i.e. is better, than what Algorithm 2 computed, which contradicts the optimality of Algorithm 2.

On the runtime: we represent lines by their end vertices. On a tree, this is enough to unambiguously define them. The invocation of Algorithm 2 can be done in $\mathcal{O}(\deg(v)^3)$. Since L_v has at most n elements, the for-loop in line 7 iterates at most n^2 times. Every operation inside the for-loop takes constant time and we can bound the total loop runtime by $\mathcal{O}(n^2)$. Overall, an iteration of the outer for-loop on a vertex v takes $\mathcal{O}(\deg(v) n^2)$. Using the fact that on a tree $\sum_{v \in V} \deg(v) = 2n - 2$, the total runtime of the algorithm is $\mathcal{O}\left(\sum_{v \in V} \deg(v)^3 + n^2 \sum_{v \in V} \deg(v)\right) = \mathcal{O}(n^3)$. ◀